Assignment 09

(5 points)

Association Rule Analysis

(2.5 points) In this question you program Association Rule Analysis to find rules X $\rightarrow$ Y, where X and Y are itemsets. Association Rule Mining algorithms in general take 2 steps:

1. Frequent Itemset Generation: Generate all itemsets of which support ≥ minsup (here minsup is the minimum support).

2. Rule Generation: Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset.

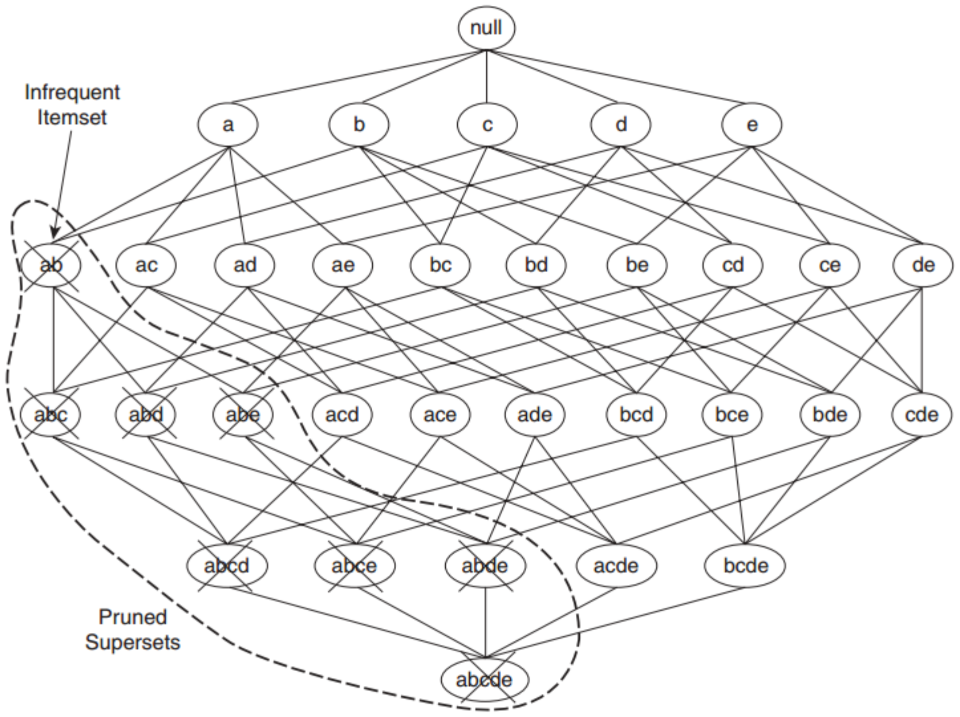


Fig. 1: Illustration of support-based pruning: if {a,b} is infrequent, all the supersets are infrequent (Introduction to Data Mining (Pang-Ning Tan et al.)

Support (s) is defined as the fraction of transactions that contain both X and Y, and confidence (c) is the measures how often items in Y appear in transactions that contain X:

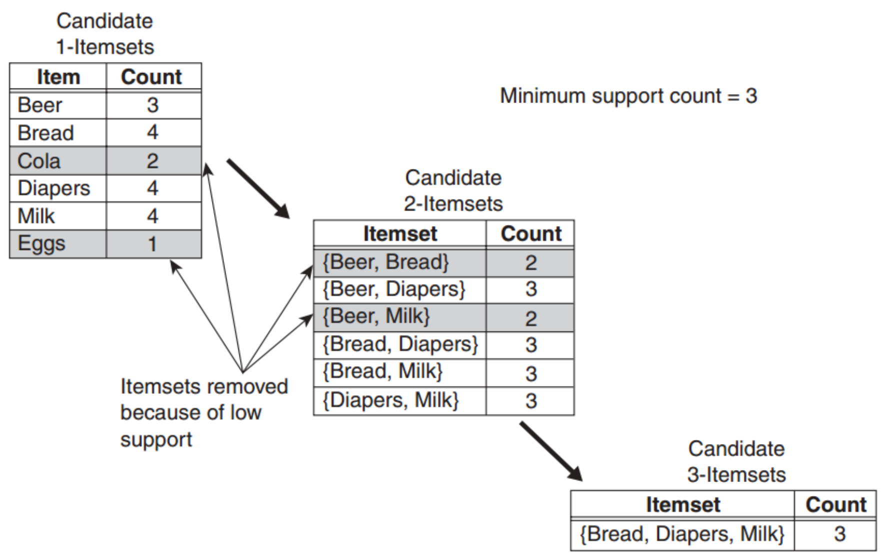
$$Support,\ s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{T}$$

$$Confidence,\ c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Apriori principle is used to reduce the number of candidate itemsets without counting support values. If an itemset is frequent, then all of its subsets must also be frequent, and if an itemset is found infrequent, all the supersets are infrequent as well. In Fig. 1, as {a,b} is found infrequent, all the supersets are infrequent as well.

The Apriori algorithm is as follows (Fig. 2). It uses support-based pruning to remove itemsets that have low support.

1. Let k=1
2. Generate frequent itemsets of length 1 (every item is a 1-itemset)
3. Repeat until no new frequent itemsets are identified
    - Generate length (k+1) candidate itemsets from length k frequent itemsets
    - Prune candidate itemsets containing subsets of length k that are infrequent
    - Count the support of each candidate by scanning the database
    - Eliminate candidates that are infrequent, leaving only those that are frequent


Minimum support count = 3
Itemsets removed because of low support

Candidate 1-Itemsets

Item	Count
Beer	3
Bread	4
Cola	2
Diapers	4
Milk	4
Eggs	1

Candidate 2-Itemsets

Itemset	Count
{Beer, Bread}	2
{Beer, Diapers}	3
{Beer, Milk}	2
{Bread, Diapers}	3
{Bread, Milk}	3
{Diapers, Milk}	3

Candidate 3-Itemsets

Itemset	Count
{Bread, Diapers, Milk}	3

Fig. 2: Illustration of frequent itemset generation using Apriori algorithm (Introduction to Data Mining (Pang-Ning Tan et al.)

To efficiently generate rules from frequent itemsets, notice that confidence of rules generated from the same itemset, for example with L = {A,B,C,D}:

c(BCD → A) ≥ c(BC → AD) ≥ c(B → ACD)

If c(BCD → A) is found low confident, then all others above are low in confidence as well.

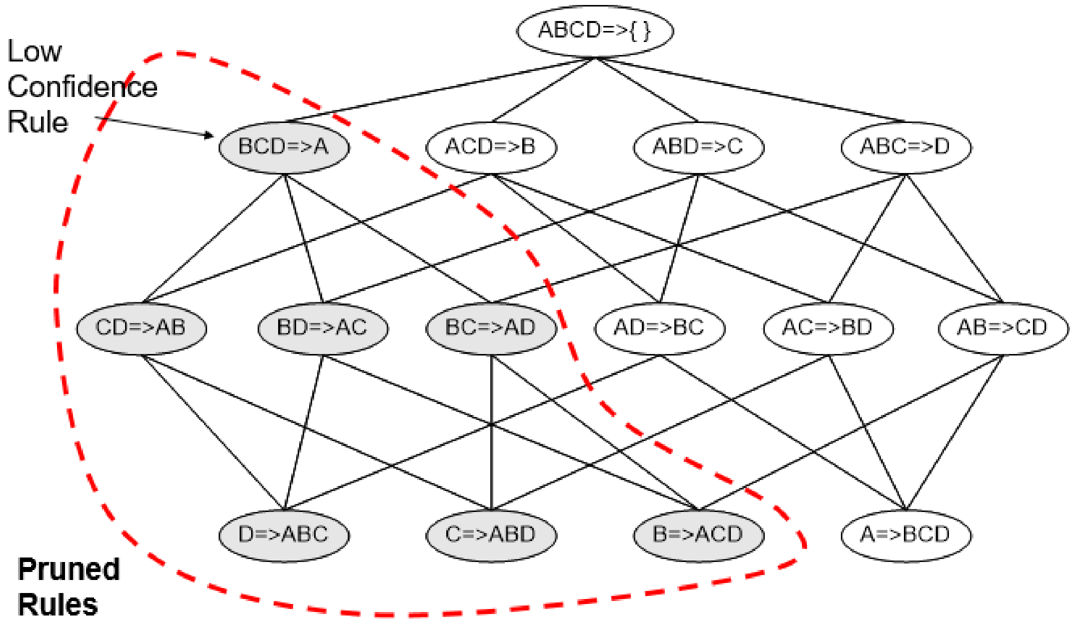


Fig. 3: Rule Generation for Apriori Algorithm (Pang-Ning Tan et al.)

The foodmart database was originally in MS Access format. The transactions are created based on sales_fact_1998 table, into the much simpler format in a text file namely "basket". The product table is exported to an Excel file under the same name for reference.

Running the whole transaction set may take long (supermarket.py). Examples below are from running a small set of 20 transactions, using minimum support of 0.05, and confidence of 0.9 (supermarket_subset.py)

```
frozenset([' 314', ' 943']) ---> frozenset(['1146', ' 4']) conf: 1.0

frozenset([' 807', '1239']) ---> frozenset([' 1456', ' 1173']) conf: 1.0

frozenset([' 1418', ' 986', '520']) ---> frozenset([' 672', ' 1056']) conf: 1.0
```

Checking the product names in the product Excel file, they are:

[Excellent Strawberry Drink, Fabulous Diet Soda] → [Tri-State Cantelope, Washington Cream Soda]

[Ebony Potatos, Plato Apple Butter] → [Hermanos Sweet Onion, Horatio Cheese Dip]

[Gauss Monthly Home Magazine, Even Better String Cheese, Tell Tale Honey Dew] → [Gorilla Havarti Cheese, Bird Call Conditioning Shampoo]

The code and examples were provided by third-party at

http://aimotion.blogspot.com/2013/01/machine-learning-and-data-mining.html

a. (2 points) Read, understand, and run the Python code.

Pick either R or MATLAB to do the work based on the working Python code is given to you. 1 additional point if you write code in both MATLAB, and R. Logics, variable names, and function names should follow the ones in the given Python code as much as possible (You will write the module Apriori as well as functions in either R or MATLAB).

b. (0.5 point) Explain your code (you can do this line by line, or small blocks of code).

Anomaly Detection

(2.5 point) Use the fraud example data set accompanying the assignment. There are 2 normal variables. The algorithm identifies the outliers (fraudulent transactions). In the given code, data are read from the Excel workbook. 60% of data are used for estimating an epsilon value such that an example x with probability $p(x,\mu,\Sigma)<\varepsilon$ is considered an outlier.

ε is selected based on the F1 score, which is

$$F1 = 2TP/(2TP + FP + FN)$$

Such that highest F1 score is highest.

In the formula:

TP: True positive

FP: False positive

FN: False negative

ε is then used to find the outliers in the rest of the data in X. The accuracy is about 98.25%

a. (2 points) Read, understand, and run the MATLAB code.

Pick either R or Python to do the work based on the working MATLAB code is given to you. 1 additional point if you write code in both Python, and R. Logics, variable names, and function names should follow the ones in the given MATLAB code as much as possible.

b. (0.5 point) Explain your code (you can do this line by line, or small blocks of code).

Submission: your script, and a document explaining your work in details to show your understanding of the theory behind the code, not just coding.

Note: in problem 1, another the dataset contains the closing stock prices for S&P500 stocks for a period of time. Their symbols show on the column headers. The companies operate in 10 sectors as follows (from SP500Companies.xls):

Health Care
Financials
Information Technology
Industrials
Utilities
Materials
Consumer Staples
Consumer Discretionary
Energy
Telecommunications Services

In the preprocessing step, a new data set is created to indicate if the stock prices increase compared with the previous day (1 or 0 corresponding to UP or DOWN).

Running Association Rule Mining, the resulted rules can be something like

[HCBK_UP] → [MTB_UP]

[ACE_UP] → [CB_UP]

[UNM_UP] → [LNC_UP]

The code wasn't tested and was provided just for reference so that you know the idea of using the algorithm for stock prices (stockmove.py, and stockmove_subset.py).